Bandwidth allocation method for fixed wireless networks

Dat Ton, Kamlesh Rath
Cowave Networks
47224 Mission Falls Court
Fremont, CA 94539

Abstract – A simple, fair, good-performance bandwidth allocation algorithm for wireless networks is presented. Using matrix of interlink interference and a list of links' bandwidth requests, the algorithm can schedule link activities to obtain non-collision transmissions. All bandwidth requests are served fairly and near-optimally based on the heuristic algorithm. Bandwidth granted for each link is prorated based on its requested bandwidth, total requested bandwidth in the network, and network capacity. The algorithm can be used for centralized bandwidth allocation and works with any network topology including mesh.

1. Introduction

This paper describes the problem of optimizing bandwidth allocation for a fixed wireless network. It proposes a simple centralized algorithm to create a fair, efficient, high-performance bandwidth allocation schedule.

Bandwidth allocation is based on request of each individual link, all the bandwidth requests in the network, link capacity, and inter-link interference. Knowledge of the whole network, in terms of interference, eligible links, and link bandwidth requests are needed to create the schedule. Only one entity, called hub in this paper, needs this global view to compute the schedule. The schedule does not contain the actual slot assignment but each node will compute it using a deterministic algorithm. Each grant in the schedule is a small integer. The schedule is not required to be sent periodically. The schedule is sent only when it changes. Hence, only a small amount of control traffic is used for dispatching the schedule.

2. Network model

A network is defined as a set of links between nodes. For example, the uni-directional link between Node I and Node J is called l_{ij}. Suppose that there are N nodes and M directional links (l_{ij} and l_{ji} are considered different links) in a network.

The interference between links in the network determines which links in the network can operate simultaneously. In other words, if a link l_{ij} is active, there exist a set of links L_{ij} which cannot be active at the same time. The set of all L_{ij} in the network constitute the *interference matrix* I of the network. Let us define the *degree of interference* $\alpha(l_{ij}, L)$ of a directional link l_{ij} in a set L of links as the number of links in set L that cannot be active due to interference while link l_{ij} is active.

We represent the bandwidth needed by links to carry actual traffic over a specific time period as a set of link bandwidth requests. The request may be zero. In that case, no traffic is to be carried over the link. Since link capacities vary depending on various link parameters, bandwidth requests are expressed in unit of *credits*, not bps (bits/sec). Credit is a unit the resource bandwidth allocation algorithm uses to maintain fair bandwidth distribution between links. It is the result of normalization of requested bandwidth (in terms of bps) with respect to the corresponding link capacity.

Example:
Figure 1 shows a network with 11 nodes and 20 directional links: $\{ l_{0,1}, l_{1,0}, l_{0,2}, l_{2,0}, l_{1,3}, l_{3,1}, l_{1,4}, l_{4,1}, l_{2,5}, l_{5,2}, l_{2,6}, l_{6,2}, l_{2,7}, l_{7,2}, l_{4,8}, l_{8,4}, l_{5,9}, l_{95}, l_{6,10}, l_{10,6} \}$
Suppose the set of links $L_{0,1}$ that gets interference (in other words, cannot be active) while link $l_{0,1}$ is active is
$L_{0,1} = \{ l_{1,0}, l_{0,2}, l_{2,0}, l_{1,3}, l_{3,1}, l_{1,4}, l_{4,1}, l_{5,9}, l_{8,4} \}$
Similarly, suppose we have the following interference sets
$L_{1,0} = \{ l_{0,1}, l_{0,2}, l_{2,0}, l_{1,3}, l_{3,1}, l_{1,4}, l_{4,1}, l_{9,5}, l_{4,8} \}$
$L_{0,2} = \{ l_{1,0}, l_{0,1}, l_{2,0}, l_{2,5}, l_{5,2}, l_{2,6}, l_{6,2}, l_{2,7}, l_{7,2}, l_{6,10} \}$
$L_{2,0} = \{ l_{1,0}, l_{0,1}, l_{0,2}, l_{2,5}, l_{5,2}, l_{2,6}, l_{6,2}, l_{2,7}, l_{7,2}, l_{10,6} \}$
$L_{1,3} = \{ l_{3,1}, l_{1,4}, l_{4,1}, l_{0,1}, l_{1,0} \}$
$L_{3,1} = \{ l_{1,3}, l_{1,4}, l_{4,1}, l_{0,1}, l_{1,0} \}$
$L_{1,4} = \{ l_{4,1}, l_{1,3}, l_{3,1}, l_{1,0}, l_{0,1}, l_{4,8}, l_{8,4}, l_{2,5}, l_{7,2} \}$
$L_{4,1} = \{ l_{1,4}, l_{1,3}, l_{3,1}, l_{1,0}, l_{0,1}, l_{4,8}, l_{8,4}, l_{5,2}, l_{2,7} \}$
$L_{2,5} = \{ l_{5,2}, l_{0,2}, l_{2,0}, l_{2,6}, l_{6,2}, l_{2,7}, l_{7,2}, l_{5,9}, l_{9,5}, l_{1,4} \}$
$L_{5,2} = \{ l_{2,5}, l_{0,2}, l_{2,0}, l_{2,6}, l_{6,2}, l_{2,7}, l_{7,2}, l_{5,9}, l_{9,5}, l_{4,1} \}$
$L_{2,6} = \{ l_{6,2}, l_{0,2}, l_{2,0}, l_{2,5}, l_{5,2}, l_{2,7}, l_{7,2}, l_{6,10}, l_{10,6} \}$
$L_{6,2} = \{ l_{2,6}, l_{0,2}, l_{2,0}, l_{2,5}, l_{5,2}, l_{2,7}, l_{7,2}, l_{6,10}, l_{10,6} \}$
$L_{2,7} = \{ l_{7,2}, l_{0,2}, l_{2,0}, l_{2,5}, l_{5,2}, l_{2,6}, l_{6,2}, l_{4,1} \}$
$L_{7,2} = \{ l_{2,7}, l_{0,2}, l_{2,0}, l_{2,5}, l_{5,2}, l_{2,6}, l_{6,2}, l_{1,4} \}$
$L_{4,8} = \{ l_{8,4}, l_{1,4}, l_{4,1}, l_{1,0} \}$
$L_{8,4} = \{ l_{4,8}, l_{1,4}, l_{4,1}, l_{0,1} \}$
$L_{5,9} = \{ l_{9,5}, l_{2,5}, l_{5,2}, l_{0,1} \}$
$L_{9,5} = \{ l_{5,9}, l_{2,5}, l_{5,2}, l_{1,0} \}$
$L_{6,10} = \{ l_{10,6}, l_{2,6}, l_{6,2}, l_{0,2} \}$
$L_{10,6} = \{ l_{6,10}, l_{2,6}, l_{6,2}, l_{2,0} \}$

Equivalently, we can express the interference using interference matrix I:

	$l_{0,1}$	$l_{1,0}$	$l_{0,2}$	$l_{2,0}$	$l_{1,3}$	$l_{3,1}$	$l_{1,4}$	$l_{4,1}$	$l_{2,5}$	$l_{5,2}$	$l_{2,6}$	$l_{6,2}$	$l_{2,7}$	$l_{7,2}$	$l_{4,8}$	$l_{8,4}$	$l_{5,9}$	$l_{9,5}$	$l_{6,10}$	$l_{10,}$
$l_{0,1}$		1	1	1	1	1	1	1								1	1			
$l_{1,0}$	1		1	1	1	1	1	1							1			1		
$l_{0,2}$	1	1		1					1	1	1	1	1	1					1	
$l_{2,0}$	1	1	1						1	1	1	1	1	1						1
$l_{1,3}$	1	1				1	1	1												
$l_{3,1}$	1	1			1		1	1												
$l_{1,4}$	1	1			1	1		1	1					1	1	1				
$l_{4,1}$	1	1			1	1	1			1			1		1	1				
$l_{2,5}$			1	1			1			1	1	1	1	1			1	1		
$l_{5,2}$			1	1				1	1		1	1	1	1			1	1		
$l_{2,6}$			1	1					1	1		1	1	1					1	1
$l_{6,2}$			1	1					1	1	1		1	1					1	1
$l_{2,7}$			1	1				1	1	1	1	1		1						
$l_{7,2}$			1	1			1		1	1	1	1	1							
$l_{4,8}$		1					1	1								1				
$l_{8,4}$	1						1	1							1					
$l_{5,9}$	1								1	1								1		
$l_{9,5}$		1							1	1							1			
$l_{6,10}$			1								1	1								1
$l_{10,6}$				1							1	1							1	

Number 1 in the matrix shows that links in corresponding row and column cannot be active at the same time. Empty boxes in the matrix represent 0s

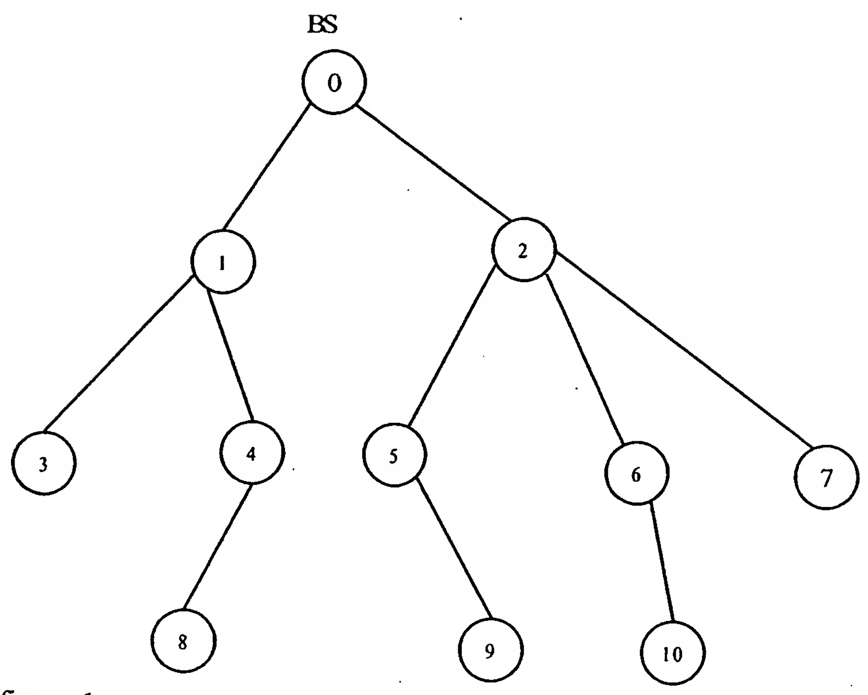

figure 1

A Link Bandwidth request is expressed in terms of the link capacity. Suppose that 64 credits are equivalent to full link capacity. If a link is given 64 credits, that link can be active all the time. If a link is given 32 credits, that link is active 50% of the time. Suppose at a particular time, there exist the following bandwidth requests (in credits):

$R_{0,2} = 35$

$R_{2,5} = 20$

$R_{2,6} = 15$

$R_{5,9} = 10$

$R_{6,10} = 10$

$R_{3,1} = 10$

$R_{1,0} = 10$

$R_{0,1} = 5$

The set of links requesting bandwidth is $L = \{ l_{0,2}, l_{2,5}, l_{2,6}, l_{5,9}, l_{6,10}, l_{3,1}, l_{1,0}, l_{0,1} \}$
Using the interference matrix I, we can compute the degree of interference of $l_{0,2}$ in this set as follows:

$\alpha(l_{0,2}, L) = I[l_{0,2}][l_{2,5}] + I[l_{0,2}][l_{2,6}] + I[l_{0,2}][l_{5,9}] + I[l_{0,2}][l_{6,10}] + I[l_{0,2}][l_{3,1}] + I[l_{0,2}][l_{1,0}] + I[l_{0,2}][l_{0,1}] = 5$

We compute other degrees of interference similarly:

$\alpha(l_{2,5}, L) = I[l_{2,5}][l_{0,2}] + I[l_{2,5}][l_{2,6}] + I[l_{2,5}][l_{5,9}] + I[l_{2,5}][l_{6,10}] + I[l_{2,5}][l_{3,1}] + I[l_{2,5}][l_{1,0}] + I[l_{2,5}][l_{0,1}] = 3$

$\alpha(l_{2,6}, L) = I[l_{2,6}][l_{0,2}] + I[l_{2,6}][l_{2,5}] + I[l_{2,6}][l_{5,9}] + I[l_{2,6}][l_{6,10}] + I[l_{2,6}][l_{3,1}] + I[l_{2,6}][l_{1,0}] + I[l_{2,6}][l_{0,1}] = 3$

$\alpha(l_{5,9}, L) = I[l_{5,9}][l_{0,2}] + I[l_{5,9}][l_{2,5}] + I[l_{5,9}][l_{2,6}] + I[l_{5,9}][l_{6,10}] + I[l_{5,9}][l_{3,1}] + I[l_{5,9}][l_{1,0}] + I[l_{5,9}][l_{0,1}] = 2$
$\alpha(l_{6,10}, L) = I[l_{6,10}][l_{0,2}] + I[l_{6,10}][l_{2,5}] + I[l_{6,10}][l_{2,6}] + I[l_{6,10}][l_{5,9}] + I[l_{6,10}][l_{3,1}] + I[l_{6,10}][l_{1,0}] + I[l_{6,10}][l_{0,1}] = 2$
$\alpha(l_{3,1}, L) = I[l_{3,1}][l_{0,2}] + I[l_{3,1}][l_{2,5}] + I[l_{3,1}][l_{2,6}] + I[l_{3,1}][l_{5,9}] + I[l_{3,1}][l_{6,10}] + I[l_{3,1}][l_{1,0}] + I[l_{3,1}][l_{0,1}] = 1$
$\alpha(l_{1,0}, L) = I[l_{1,0}][l_{0,2}] + I[l_{1,0}][l_{2,5}] + I[l_{1,0}][l_{2,6}] + I[l_{1,0}][l_{5,9}] + I[l_{1,0}][l_{6,10}] + I[l_{1,0}][l_{3,1}] + I[l_{1,0}][l_{0,1}] = 3$
$\alpha(l_{0,1}, L) = I[l_{0,1}][l_{0,2}] + I[l_{0,1}][l_{2,5}] + I[l_{0,1}][l_{2,6}] + I[l_{0,1}][l_{5,9}] + I[l_{0,1}][l_{6,10}] + I[l_{0,1}][l_{3,1}] + I[l_{0,1}][l_{1,0}] = 4$

3. Problem formulation

We assume that TDMA is used to multiplex links activities. Given the constraints of the interference matrix and a list of bandwidth requests, we attempt to find a schedule to optimally make use of total network capacity and fairly satisfy bandwidth requests.

An equivalent problem is to find an optimal schedule that satisfies all requests using the least amount of network resources, in this case, credits or time. If we define the *average activity concurrency* as the average number of concurrent active links of a schedule, the optimal schedule is the one with highest average activity concurrency.

A schedule specifies when a set of links are active and the members of the set.
Mathematically, a schedule S can be expressed as
$S = \{(L_i, G_i) \mid G_i$ is the credits assigned to set of links L_i,
L_i is the set of links that can be all active at the same time without interfering with each other $\}$

Example (cont):
Continuing on with the previous example, here is one possible schedule for links requesting bandwidth:
$S = \{ (\{ l_{5,9}, l_{6,10}, l_{3,1} \}, 10), (\{ l_{0,2} \}, 35), (\{ l_{2,6}, l_{0,1} \}, 5), (\{l_{2,6}, l_{1,0} \}, 10), (\{ l_{2,5} \}, 20) \}$
.This schedule uses 10+35+5+10+20 = 80 credits to satisfy 35+20+15+10+10+10+10+5 = 115 requested credits. The average activity concurrency is 115/80 = 1.4375
This schedule is not necessarily the best schedule for this example. In fact, using the algorithm described in the next section, we will find a better schedule using less credits while still satisfying all bandwidth requests.

An optimal schedule must satisfy the following conditions:
For any link, granted Credits equals requested credits
$\sum_{l_{jk} \subset L_i} G_i = R_{jk}$

Minimal total network resource spent
$(\sum G_i) <= (\sum G'_i)$ for $\forall$ $S' = \{(L'_i, G'_i) \}$

Since this problem is NP-hard, we present a heuristic algorithm in this paper for near optimal solution. Simulations show that in many cases it generates optimal schedules; and in cases that it does not, the schedules are usually close to optimal and are always better than average.

4. Bandwidth allocation algorithm

The algorithm is based on the assumption that there exists a centralized node (hub) in the network that coordinates all network activities. This hub keeps the following data structures to represent its knowledge of the network:

- Interference matrix (defined above). It is important to note that interference matrix is symmetrical.
- Topology matrix: defines valid links that can transmit/receive data. This is a proper subset of the interference matrix.
- A list of credit request tokens. Each token represents a directional link that needs bandwidth.

Let us assume that each node in the network conveys its knowledge of interference, topology and its bandwidth needs to the hub. The mechanism on how this information is transported to the hub is out of the scope of this paper. The hub collects this information from individual nodes and constructs the interference matrix, topology matrix and list of credit tokens to have a complete view of the network.

Bandwidth allocation algorithm running at hub is described as followed:
1. Sort credit request tokens in the descending order of the product of requested credits and degree of interference $\alpha(l_{ij}, L)$, where L is the set of links requesting for credits.

2. Pick the first token (having largest product). This is the first candidate of the set of links to be allocated credit for this round. Eliminate all other tokens from this round who cannot be active due to this link's activity.

3. Walk down the list and pick the next eligible token. This is the second candidate of the set of links to be allocated credits for this round. Eliminate all other tokens from this round who cannot be active due to this link's activity. Continue this step until the list is exhausted.

4. The result is a set of links that can be active at the same time $L_1 = \{ l_1, l_2, \ldots, l_n\}$. Let β_{li} be requested credits of link l_i. Amount of credits allocated to each element of set L_1 will be $\gamma_1 = \min\{\beta_{l1}, \beta_{l2}, \ldots, \beta_{ln}\}$. Adjust requested credits for every element in L_1: $\beta_{li} = \beta_{li} - \gamma_1$. Remove token(s) which having zero requested credits from the list of tokens.

5. Adjust degree of interference of affected links (due to the fact that some tokens have been removed)

6. Repeat steps 1-5 until the list of tokens is empty.

7. The result is a list of (L_1, γ_1), (L_2, γ_2) …. (L_k, γ_k). Now we can prorate this list to attain the final schedule. Let S be the total resource of the network in terms of credit; and let $\chi_i = \gamma_I * S // \Sigma^{0,k} \gamma_j$. The list (L_1, χ_1), (L_2, χ_2) …. (L_k, χ_k) represents how the links are organized into sets of concurrent active links and how much resource each set of links are supposed to get. This list is broadcast to all nodes in the network.

Example (cont):
Let us now use this algorithm to compute the schedule for our previous example.

Step 1:

Link	Degree of interference $\alpha(l_{ij}, L)$	Requested credit R_{ij}	$\alpha(l_{ij}, L) * R_{ij}$
$l_{0,2}$	5	35	175
$l_{2,5}$	3	20	60
$l_{2,6}$	3	15	45
$l_{1,0}$	3	10	30
$l_{5,9}$	2	10	20
$l_{6,10}$	2	10	20
$l_{3,1}$	2	10	20
$l_{0,1}$	4	5	20

Steps 2-5:
We get the first Schedule S = { ({$l_{0,2}$, $l_{5,9}$, $l_{3,1}$}, 10) }

Go back to step 1:

Link	Degree of interference $\alpha(l_{ij}, L)$	Requested credit R_{ij}	$\alpha(l_{ij}, L) * R_{ij}$
$l_{0,2}$	5	25	125
$l_{2,5}$	2	20	40
$l_{2,6}$	2	15	30
$l_{1,0}$	2	10	20
$l_{6,10}$	2	10	20
$l_{0,1}$	2	5	10

Steps 2-5:
We get a revised Schedule S = { ({$l_{0,2}$, $l_{5,9}$, $l_{3,1}$}, 10), ({$l_{0,2}$}, 25) }

Go back to step 1:

Link	Degree of interference $\alpha(l_{ij}, L)$	Requested credit R_{ij}	$\alpha(l_{ij}, L) * R_{ij}$
$l_{2,5}$	1	20	20
$l_{2,6}$	1	15	15
$l_{1,0}$	1	10	10

$l_{6,10}$	1	10	10
$l_{0,1}$	1	5	5

Steps 2-5:
We get a revised Schedule S = { ({$l_{0,2}$, $l_{5,9}$, $l_{3,1}$}, 10), ({$l_{0,2}$}, 25), ({$l_{2,5}$, $l_{1,0}$, $l_{6,0}$ }, 10)}

Go back to step 1:

Link	Degree of interference $\alpha(l_{ij}, L)$	Requested credit R_{ij}	$\alpha(l_{ij}, L) * R_{ij}$
$l_{2,6}$	1	15	15
$l_{2,5}$	1	10	10
$l_{0,1}$	0	5	0

Steps 2-5:
We get a revised Schedule
S = { ({$l_{0,2}$, $l_{5,9}$, $l_{3,1}$}, 10), ({$l_{0,2}$}, 25), ({$l_{2,5}$, $l_{1,0}$, $l_{6,0}$}, 10) , ({$l_{2,6}$, $l_{0,1}$}, 5) }

Go back to step 1:

Link	Degree of interference $\alpha(l_{ij}, L)$	Requested credit R_{ij}	$\alpha(l_{ij}, L) * R_{ij}$
$l_{2,6}$	1	10	10
$l_{2,5}$	1	10	10

Steps 2-5:
We get a revised Schedule
S = { ({$l_{0,2}$, $l_{5,9}$, $l_{3,1}$}, 10), ({$l_{0,2}$}, 25), ({$l_{2,5}$, $l_{1,0}$, $l_{6,0}$}, 10) , ({$l_{2,6}$, $l_{0,1}$}, 5) , ({$l_{2,6}$}, 10) }

Go back to step 1:

Link	Degree of interference $\alpha(l_{ij}, L)$	Requested credit R_{ij}	$\alpha(l_{ij}, L) * R_{ij}$
$l_{2,5}$	1	10	10

Steps 2-5:
We get the schedule
S = { ({$l_{0,2}$, $l_{5,9}$, $l_{3,1}$}, 10), ({$l_{0,2}$}, 25), ({$l_{2,5}$, $l_{1,0}$, $l_{6,0}$}, 10) , ({$l_{2,6}$, $l_{0,1}$}, 5) , ({$l_{2,6}$}, 10) , ({$l_{2,5}$}, 10) }

This schedule uses 10+25+10+5+10+10 = 70 credits to satisfy 35+20+15+10+10+10+10+5 = 115 requested credits. The average activity concurrency is 115/70 = 1.6428. Obviously, this schedule is better than the one that we had in the previous example. In fact, we can prove that this schedule is the optimal one for this particular example. There is no other schedule that can use less number of credits to satisfy all these bandwidth requests.

Step 7:

Since our total resource is only 64 credits, the previous schedule is prorated to obtain the final schedule:
$S_f = \{ (\{l_{0,2}, l_{5,9}, l_{3,1}\}, 9), (\{l_{0,2}\}, 23), (\{l_{2,5}, l_{1,0}, l_{6,0}\}, 9), (\{l_{2,6}, l_{0,1}\}, 5), (\{l_{2,6}\}, 9), (\{l_{2,5}\}, 9) \}$
This schedule is broadcast to all nodes in the network.

Upon receiving the schedule, each node in the network uses the binary allocation map scheme to compute its own slot assignment. Allocation map is an array of numbers that is used to map a range of consecutive numbers to partially equally spaced numbers. The idea is that, given a portion of resources, a node can figure out its active timeslots by projecting that portion (consecutive numbers) through the map. For example, all links in set L_i are assigned to the range $[\Sigma^{0,i-1} \chi_j , \Sigma^{0,i} \chi_j]$, which, in turn, will represent a set of near-equally spaced time slots.

Example (cont):
Let us assume that our allocation map is designed for 64 time slots, corresponding to 64 credits.

Allocation map for 64 time slots

tslot	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
index	1	33	17	49	9	41	25	57	5	37	21	53	13	45	29	61

tslot	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
index	3	35	19	51	11	43	27	59	7	39	23	55	15	47	31	63

tslot	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
index	2	34	18	50	10	42	26	58	6	38	22	54	14	46	30	62

tslot	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
index	4	36	20	52	12	44	28	60	8	40	24	56	16	48	32	64

A range of credit indices can be deduced for each set of links in the final schedule S_f. For example, the set $\{l_{0,2}, l_{5,9}, l_{3,1}\}$ is correspondent to [1,9]. Set $\{l_{0,2}\}$ is correspondent to [10,32]; and so on.

$S_f = \{ (\{l_{0,2}, l_{5,9}, l_{3,1}\}, 9), (\{l_{0,2}\}, 23), (\{l_{2,5}, l_{1,0}, l_{6,0}\}, 9), (\{l_{2,6}, l_{0,1}\}, 5), (\{l_{2,6}\}, 9), (\{l_{2,5}\}, 9) \}$

Using the combination of allocation map and the final schedule S_f, any node is aware of which link is active at a particular time slot t. For example, the set $\{l_{0,2}, l_{5,9}, l_{3,1}\}$ is active in time slots 1, 5, 9, 17, 25, 33, 41, 49, 57.

5. Maximizing network capacity using unscheduled time slots

To facilitate the explanation of using unscheduled time slots, let us use the schedule obtained in previous example.

S = { ({$l_{0,2}$, $l_{5,9}$, $l_{3,1}$}, 10), ({$l_{0,2}$}, 25), ({$l_{2,5}$, $l_{1,0}$, $l_{6,0}$}, 10) , ({$l_{2,6}$, $l_{0,1}$}, 5) , ({$l_{2,6}$}, 10) , ({$l_{2,5}$}, 10) }

Some notable points need to be made about this schedule:

1. The number of links in each set tends to be highest at the beginning of the schedule and tends to taper off toward the end of the schedule.
2. Even with the set causing most interference in the network, there will be some links that can be active at the same time without causing interference to the links in the set.
3. The interference caused by sets at the beginning of the schedule tend to be the highest; and that interference tends to taper off going toward the end of the schedule.

With these observations, we can see the scheduled bandwidth very likely represents only about half of total network capacity. Hence, collision-based mechanism is devised to use the other half, which is going to be wasted if not used otherwise.

Each node in the network maintains, for each of its local links, one set of links interfering with that link. Local links are links directly connected to the node. By using the schedule S broadcast by the Hub, a node knows which of its local links can be active without interfering with the scheduled links which are currently active. An active unscheduled link at time slot *t* is a link that is not scheduled to be active at time *t*, but could be made active if the intended receiver is ready to receive. This can be decided by its directly connected nodes because this activity does not cause interference with the current active scheduled links. A link can be unscheduled at one time slot and is scheduled in another time slot. Active unscheduled links can interfere and collide which each other, but they do not interfere with the currently active scheduled links.

Unscheduled links are mainly used when a node does not have uplink scheduled bandwidth and need to request bandwidth or need to send some small uplink transient traffic. It is used to boost up network capacity as well as network response time.

Example (cont):
Using our previous example, the final schedule is
S = { ({$l_{0,2}$, $l_{5,9}$, $l_{3,1}$}, 10), ({$l_{0,2}$}, 25), ({$l_{2,5}$, $l_{1,0}$, $l_{6,0}$}, 10) , ({$l_{2,6}$, $l_{0,1}$}, 5) , ({$l_{2,6}$}, 10) , ({$l_{2,5}$}, 10) }

Let pick one time slot t. Suppose that it corresponds to ({$l_{2,6}$, $l_{0,1}$}, 5) in the schedule. It means that $l_{2,6}$ and $l_{0,1}$ are active at time slot t. The matrix of interference tells us that any of links { $l_{4,8}$ $l_{9,5}$ } can also be active. Although each node does not maintain the matrix of interference for the whole network, it does keep sets of interference links for each of its local link. Hence local nodes (4 and 9) know that they can activate the link at time slot t. In this specific example, if both $l_{4,8}$ $l_{9,5}$ are active, they still don't collide. However, that is not always the case. Nodes will use backoff mechanism to resolve collision if it happens.